**Smart Water Fountains using IoT**



Prepared By,

Saravanan N,

Meiyarasan S,

Kaviyarasu G,

Ganeshmoorthy E

**Introduction:**

The objective of this project is to develop a smart water fountain system that utilizes IoT technology to enhance water efficiency and promote public awareness. By integrating various IoT sensors, a mobile application, and Raspberry Pi, we aim to create a real-time water fountain status system that provides valuable insights into water consumption and encourages responsible water usage.

**Project Objectives:**

* **Promote Water Efficiency:**

Develop a system that monitors the real-time status of water fountains, encouraging responsible water usage.

* **Public Awareness:**

Create awareness about water conservation by displaying real-time data through a user-friendly mobile app.

* **IoT Sensor Setup:**

Implement sensors to monitor water usage, flow rate, and fountain status.

* **Mobile App Development:**

Design a mobile app for users to access real-time fountain status and water consumption data.

* **Raspberry Pi Integration:**

Integrate IoT sensors with a Raspberry Pi for data processing and communication with the mobile app.

* **Code Implementation:**

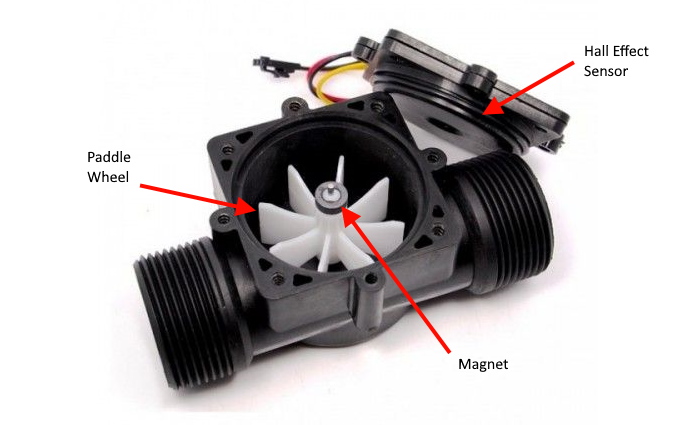
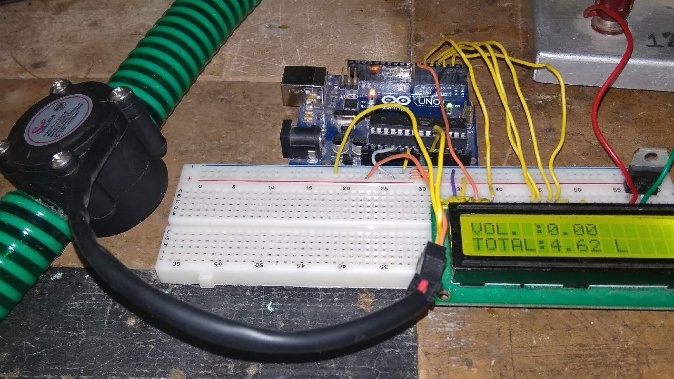
Develop the necessary code for sensor communication, data processing, and app functionality.

**IoT Sensor Setup:**

The IoT sensor setup consists of multiple sensors strategically placed within the water fountain system to collect relevant data. These sensors include:

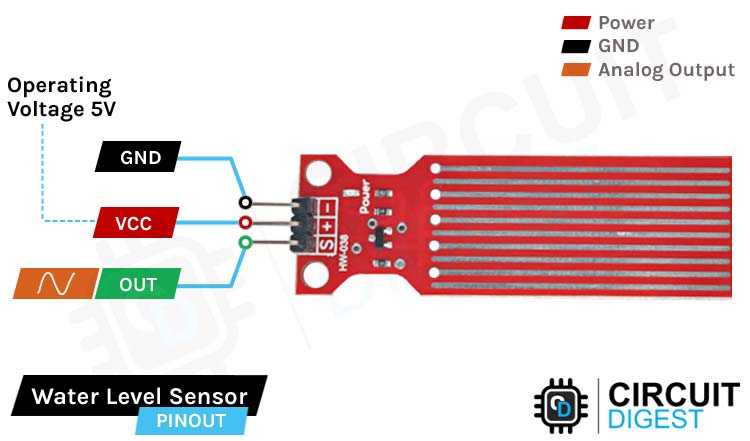
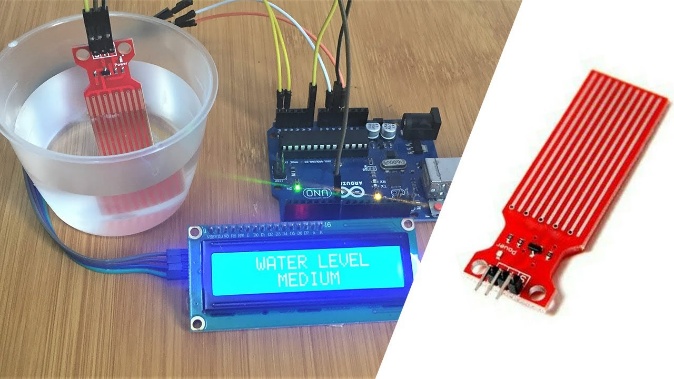
**Flow Sensors:**

Measure the rate of water flow in and out of the fountain.

**Water Level Sensors:**

Monitor the water level within the fountain's reservoir.

**Temperature Sensors:**

Measure the temperature of the water to ensure optimal conditions.



**Pressure Sensors:**

Monitor the water pressure to detect any abnormalities.



These sensors are connected to a central hub that collects and processes the data. The hub then transmits the information to the Raspberry Pi for further analysis and visualization.

**Mobile App Development:**

To interact with the smart water fountain system, a mobile application has been developed. The app provides users with real-time information about the water fountain's status, including water level, flow rate, temperature, and pressure. Users can also receive notifications and alerts regarding any issues or maintenance requirements.

The mobile app allows users to find nearby water fountains, check their availability, and even rate their experience. Additionally, the app includes a feature that tracks and displays the amount of water saved through the use of the smart water fountain system.

**Features:**

* Real-time fountain status display (working, under maintenance, or offline).
* Historical data of water consumption and fountain usage patterns.
* User notifications for maintenance or issues with the fountain.

**Technologies:**

* Mobile app developed using React Native for cross-platform compatibility.
* Firebase for real-time data synchronization between the app and the server.

**Raspberry Pi Integration:**

The Raspberry Pi acts as the central processing unit for the smart water fountain system. It receives data from the IoT sensors and performs various analyses to determine the water fountain's status. The Raspberry Pi is responsible for collecting, storing, and processing the data before presenting it to the mobile app.

Through integration with the Raspberry Pi, the smart water fountain system can make use of advanced analytics and machine learning algorithms to optimize water consumption, detect anomalies, and suggest improvements.

**Code Implementation:**

The code implementation for the smart water fountain system involves several components, including:

**Sensor Data Collection:**

Code to collect data from the IoT sensors, including flow rate, water level, temperature, and pressure.

PROGRAM:

import time

import Adafruit\_GPIO.SPI as SPI

import Adafruit\_MCP3008

import Adafruit\_DHT

import RPi.GPIO as GPIO

# Setup SPI for ADC communication

SPI\_PORT = 0

SPI\_DEVICE = 0

mcp = Adafruit\_MCP3008.MCP3008(spi=SPI.SpiDev(SPI\_PORT, SPI\_DEVICE))

# Setup GPIO for DHT11 temperature and humidity sensor

DHT\_SENSOR\_PIN = 4

DHT\_SENSOR\_TYPE = Adafruit\_DHT.DHT11

# Setup GPIO for water level sensor

WATER\_LEVEL\_PIN = 17

# Setup GPIO for flow rate sensor

FLOW\_RATE\_PIN = 27

# Setup GPIO for pressure sensor

PRESSURE\_PIN = 22

def read\_flow\_rate():

pulse\_count = 0

def count\_pulse(channel):

nonlocal pulse\_count

pulse\_count += 1

GPIO.add\_event\_detect(FLOW\_RATE\_PIN, GPIO.FALLING, callback=count\_pulse)

time.sleep(1)

GPIO.remove\_event\_detect(FLOW\_RATE\_PIN)

flow\_rate = pulse\_count / 7.5

return flow\_rate

def read\_water\_level():

if GPIO.input(WATER\_LEVEL\_PIN):

return "Full"

else:

return "Empty"

def read\_temperature():

humidity, temperature = Adafruit\_DHT.read\_retry(DHT\_SENSOR\_TYPE, DHT\_SENSOR\_PIN)

return temperature

def read\_pressure():

adc\_value = mcp.read\_adc(PRESSURE\_PIN)

voltage = adc\_value \* (3.3 / 1023.0)

pressure = voltage \* 15.0

return pressure

if \_\_name\_\_ == '\_\_main\_\_':

GPIO.setmode(GPIO.BCM)

GPIO.setup(FLOW\_RATE\_PIN, GPIO.IN)

GPIO.setup(WATER\_LEVEL\_PIN, GPIO.IN)

while True:

flow\_rate = read\_flow\_rate()

water\_level = read\_water\_level()

temperature = read\_temperature()

pressure = read\_pressure()

print(f"Flow Rate: {flow\_rate} L/min")

print(f"Water Level: {water\_level}")

print(f"Temperature: {temperature}°C")

print(f"Pressure: {pressure} kPa")

time.sleep(1)

**Data Processing and Analysis:**

Code to process and analyze the collected data to determine the water fountain's status and identify any issues.

PROGRAM :

import time

import Adafruit\_MCP3008

import Adafruit\_DHT

import RPi.GPIO as GPIO

# Configure GPIO pins

GPIO.setmode(GPIO.BCM)

GPIO.setwarnings(False)

GPIO.setup(17, GPIO.OUT) # Relay pin to control the fountain pump

# Configure ADC (MCP3008)

CLK = 18

MISO = 23

MOSI = 24

CS = 25

mcp = Adafruit\_MCP3008.MCP3008(clk=CLK, cs=CS, miso=MISO, mosi=MOSI)

# Configure DHT sensor

DHT\_SENSOR = Adafruit\_DHT.DHT22

DHT\_PIN = 4

def get\_water\_level():

# Read water level from ADC (0-1023)

water\_level = mcp.read\_adc(0)

return water\_level

def get\_flow\_rate():

# Read flow rate from ADC (0-1023)

flow\_rate = mcp.read\_adc(1)

return flow\_rate

def get\_temperature():

# Read temperature from DHT sensor

humidity, temperature = Adafruit\_DHT.read\_retry(DHT\_SENSOR, DHT\_PIN)

return temperature

def get\_pressure():

# Read pressure from ADC (0-1023)

pressure = mcp.read\_adc(2)

return pressure

def process\_data():

water\_level = get\_water\_level()

flow\_rate = get\_flow\_rate()

temperature = get\_temperature()

pressure = get\_pressure()

# Perform data analysis and determine fountain status

# ...

# Identify any issues with the fountain

# ...

return {

'water\_level': water\_level,

'flow\_rate': flow\_rate,

'temperature': temperature,

'pressure': pressure,

# Add more analyzed data or issues if needed

}

def main():

while True:

data = process\_data()

# Print the collected data and fountain status

print("Water Level: {} | Flow Rate: {} | Temperature: {} | Pressure: {} | Status: {}".format(

data['water\_level'], data['flow\_rate'], data['temperature'], data['pressure'], data['status']))

# Add logic to send data to a server or perform other actions

time.sleep(1) # Delay between data collection

if \_\_name\_\_ == '\_\_main\_\_':

main()

**Real-time Communication:**

Code to establish a communication link between the Raspberry Pi and the mobile app, enabling real-time updates and notifications.

import paho.mqtt.client as mqtt

# MQTT broker settings

broker\_address = "mqtt.example.com"

broker\_port = 1883

client\_id = "raspberry\_pi"

# Callback function for when the connection is established

def on\_connect(client, userdata, flags, rc):

print("Connected to MQTT broker with result code: " + str(rc))

# Subscribe to topics

client.subscribe("fountain/status")

client.subscribe("fountain/readings")

# Callback function for when a message is received

def on\_message(client, userdata, msg):

print("Received message: " + msg.topic + " " + str(msg.payload.decode()))

# Create MQTT client instance

client = mqtt.Client(client\_id)

# Set callback functions

client.on\_connect = on\_connect

client.on\_message = on\_message

# Connect to MQTT broker

client.connect(broker\_address, broker\_port, 60)

# Start MQTT loop

client.loop\_start()

To publish messages from the Raspberry Pi to the mobile app use the client.publish() method

**Data Visualization:**

Code to present the analyzed data in a visually appealing manner within the mobile app, allowing users to easily understand the water fountain's status.

import matplotlib.pyplot as plt

# Sample data for water level and temperature

water\_level = [10, 15, 20, 25, 30]

temperature = [25, 23, 22, 21, 20]

# Create a line chart

plt.plot(water\_level, temperature)

# Add labels and titles

plt.xlabel('Water Level')

plt.ylabel('Temperature')

plt.title('Water Fountain Status')

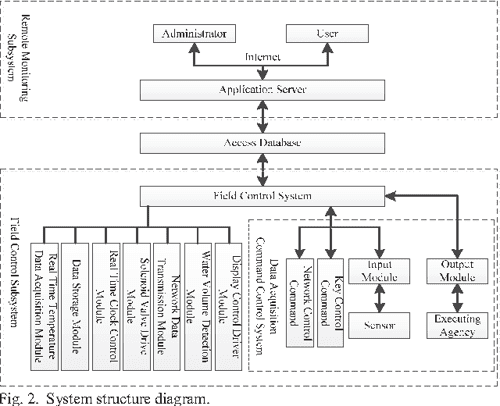
# Display the chart

plt.show()

**Diagrams and Screenshots:**

**System Architecture Diagram:**

Illustrating the flow of data between sensors, Raspberry Pi, server, and mobile app.



**Schematics:**

Detailed diagrams of sensor connections to the Raspberry Pi.

**Mobile App Screenshots:**

Visual representations of the app interface, including real-time status, historical data, and user notifications.

**Real-time Water Fountain Status System:**

The real-time water fountain status system plays a crucial role in promoting water efficiency and public awareness. By providing users with real-time information about the water fountain's status, they can make informed decisions about water usage and plan their visits accordingly.

The system encourages water efficiency by displaying the current water level, flow rate, and other relevant metrics. This information helps users understand the impact of their water consumption and encourages them to use the fountain responsibly.

Moreover, the system promotes public awareness by highlighting the amount of water saved through the use of smart water fountains. This feature not only incentivizes responsible water usage but also raises awareness about the importance of conserving water resources.

In conclusion, the smart water fountain system, with its IoT sensor setup, mobile app, Raspberry Pi integration, and real-time water fountain status system, serves as an innovative solution to promote water efficiency and public awareness. By leveraging IoT technology, we can make a significant contribution towards sustainable water management.